NWG/RFC 385
NIC 11357
Updates: RFC 354

Abhay K. Bhushan
MIT-MAC
August 18, 1972
Revd 11 Sept. 72

COMMENTS ON THE FILE TRANSFER PROTOCOL (RFC 354)

The following comments pertain to the File Transfer Protocol, NWG/RFC 354. The comments include errata, further discussion, emphasis points, and additions to the protocol. I shall incorporate these comments into the main protocol document after we have had sufficient experience.

1. Please note the following corrections:
   (i) Page 2, line 15: replace user-FTP by server-FTP.
   (ii) Page 3, line 12: replace III.A by III.C.
   (iii) Page 15, last para, line 1: replace user s by user is.
   (iv) Page 28, line 21: replace CRCRLF by CRLF.
   (v) Page 27, line 10: replace 451,451 by 451.
   (vi) Note that on Page 26, line 15 mode code is S|B|T|H.

2. The language of RFC 354 reads that it is recommended for hosts to implement the default parameters. The sense of the word recommended should be taken as required. Thus the required minimum implementation for FTP servers is:

   Type - ASCII (8-bit bytes)
   Mode - Stream
   Structure - File
   Commands - RETR, STOR, USER (and PASS), SOCK, and BYE

3. The "Print File-ASCII" and "EBCIDIC Print File" types are incorrectly specified (pages 10 and 11, RFC 354). The real problem with print files is of ASA (Fortran) vertical format control. Instead of the two print file types, there should really be three types as described below:

   EBCDIC - The sender transfers data using the EBCDIC character code and 8-bit transfer byte size. The CRLF convention is used for vertical format control. This type will be used for efficient transfer of EBCDIC files between systems which use EBCDIC for their internal character representation.

   ASCII with ASA vertical format control - This is the "Print file-ASCII" defined in RFC 354. The server is to transform the data in accordance with ASA (Fortran) vertical format control procedures for printing on printers that still use this standard. The data is to be transferred as 8-bit bytes.

EBCDIC with ASA vertical format control - This is the EBCDIC Print file" defined in RFC 354. The server is to transform the data in accordance with ASA (Fortran) vertical format control standards but using the EBCDIC character code. The data is to be transferred in 8-bit bytes.

The new types are to be denoted by symbols E for EBCDIC, P for Print file-ASCII and F for Formatted (ASA standard) EBCDIC print file. A discussion of the ASA vertical format control appears in NWG/RFC 189, Appendix C, and in Communications of the ACM, Vol 7, No. 10, p. 606, October 1964. According to the ASA vertical format control standards, the first character of a formatted record is not printed but determines vertical spacing as follows:

Character	Vertical Spacing before printing
Blank	One line
0	Two lines
1	To first line of next page
+	No advance

In addition to the above four, there are more characters (definned in Appendix C, RFC 189) which represent an IBM extension to the ASA standard.

4. A comparison of "stream" and "text" modes is in order. The advantages of "stream" mode are:
   1) The receiver need not scan the incoming bytes.
   2) it is usable with all data types.

   The disadvantages are:
   1) The EOF by closing the connection is not reliable.
   2) The EOR by ASCII CRLF is unreliable as the CRLF really may be valid data rather than an EOR. It is an EOR only if the sender and receiver have a prior agreement to that effect.

5. In the Block mode the protocol states that left-most bits not containing information should be zero. It appears that some sites have difficulty sending null bytes in the beginning of a block. Since it is really not necessary for these bytes to be zero, these bits are now defined to be "don't care" bits. crock

6. In the use of block mode it is possible for two or more conditions requiring different descriptor codes (suspected errors and either end of record or end of file) to exist simultaneously. Such a possibility may be handled by sending a separate EOR or EOF block with a zero byte count (this is allowed by the protocol). Also it should be noted that an EOF is an implicit EOR.

7. It needs to be emphasized again that the user-FTP must "listen" on the data socket prior to sending a command

requiring a file transfer. Specifically the user-FTP should not wait for a 255 reply (server data socket) before doing the "listen". (The security check may be done later, as the data connection can be closed if connection is to a socket other than that specified by the 255 reply). Although the protocol suggests that the 255 reply would be sent before making the connection, it does not guarantee that the 255 reply would arrive before the initiating RFC at the user site. The above argument also applies to receiving a close (NCP-CLS) on the data connection before receiving a reply indicating the reason for the close (note assertion on page 24, paragraph 3, RFC 354).

8. Although the protocol does not restrict closing or leaving open the data connection in Block and Text modes, it should be emphasized that the closing of the data connection, if it is to be done at all, should be done immediately after the file transfer rather than just after a new transfer command is received. This is because the server and user may have to test whether the data connection is open or not before doing a "listen" or an "init" respectively.

9. It should be emphasized again that 'Type' supersedes 'Byte', and that the TYPE command should be sent before the BYTE command.

10. It should be noted that both upper and lower case alphabetic characters are to be treated identically in the command syntax. This applies also to the symbols for type, mode, and structure. For example, 'A' and 'a' both indicate ASCII type.

11. It should be noted that in the 'LIST' command, the data transfer is over the data connection in type ASCII.

12. The following reply code is to be added:

        454 FTP: Cannot connect to your data socket.

    This is a fail response any of the commands requiring data transfer (including RETR, STOR, APPE, and LIST).

13. Rather than use the append command for sending mail files, a new command 'MLFL' (for mail file) is defined. The syntax of the mail file command is:

        MLFL <user>CRLF
        where
        <user> ::= <empty>| <NIC ident>| <sys ident>

    If the user field is empty or blank (one or more spaces), then the mail is destined for a printer or other designated place for site mail. <NIC ident> refers to the standard identification described in the NIC Directory of Network

Participants. A serving host may keep a table mapping <NIC ident> into <sys ident>. This would provide for uniform convenient usage. <sys ident> is the user's normal identification at the serving HOST. The use of <sys ident> would allow a network user to send mail to other users who do not have NIC identification but whose <sys ident> is known.

The intent of this command is to enable a user at the user site to mail data (in form of a file) to another user at the server site. It should be noted that the files to be mailed are transmitted via the data connection in ASCII type. These files should be appended to the destination user's mail by the server in accordance with serving HOST mail conventions. The mail may be marked as sent from the particular using HOST and the user specified by the 'USER' command. The reply codes for the "MLFL" command are identical to that in the "APPE" command, as shown below:

COMMAND	SUCCESS	FAIL
MLFL	250	451,454,500-506
Sec. reply	252	452,453

14. The 'MLFL' command for network mail, though a useful and essential addition to the FTP command repertoire, does not allow TIP users to send mail conveniently without using third hosts. It would be more convenient for TIP users to send mail over the TELNET connection instead of the data connection as provided by the 'MLFL' command. The following 'MAIL' command is therefore defined to send mail via the TELNET connection:

MAIL <user>CRLF

the syntax of <user> is identical to that in the MLFL command described above. After the 'MAIL' command is received, the server is to treat the following lines as text of the mail sent by the user. The mail text is to be terminated by a line containing only a single period, that is the character sequence ".CRLF" in a new line. The following new reply codes are defined to handle the mail command:

350 Enter mail, terminate by a line with only a '.'
256 Mail completed.

The reply codes are:

COMMAND	SUCCESS	FAIL
MAIL	350	450,451,500-506
Sec Reply	256	

15. An additional access control command called account (ACCT) is now defined to facilitate accounting in systems such as

TENEX which require in addition to user and password, a separate account specification. The 'ACCT' command is different from the 'PASS' command in that it is not necessarily related to the 'USER' command and may arrive at any time. For example, a user may transfer different files using different accounts. The 'ACCT' command has the same reply codes as the 'PASS' command (230 for success and 430-432,500-506 for fail). Some servers may require that an account command must be sent before the user is "logged in". For suchcases the success reply to the 'PASS' command could be '330 Enter account'.

16. Since password information is quite sensitive, it is desirable in general to "mask" it or suppress type out. It appears that the server has really no fool-proof effective way to achieve this. It is therefore the user-FTP process responsibility to hide the sensitive password information.

17. The FTP is an open-ended protocol designed for easy expandability. Experimental commands may be defined by sites wishing to implement such commands. These experimental commands should begin with the alphabetic character 'X'. Standard reply codes may be used with these commands. If new reply codes need to assigned, these should be chosen between 900 and 999. If the experimental command is useful and of general interest, it shall be included in the FTP command repertoire.